Automated Web Software Testing with Selenium

Regina Ranstrom[1,2], Cynthia Nikolai[2], Greg Madey[2]

Computer Science and Engineering, [1]Northeastern University, [2]University of Notre Dame

Introduction

There is a great responsibility for developers and testers to ensure that web software exhibits high reliability and speed. Somewhat recently, the software community has seen a rise in the usage of AJAX in web software development to achieve this goal. The advantage of AJAX applications is that they are typically very responsive. The vEOC is an Emergency Management Training application which requires this level of interactivity. Selenium is great in that it is an open source testing tool that can handle the amount of JavaScript present in AJAX applications, and even gives the tester the freedom to add their own features. Since web software is so frequently modified, the main goal for any test developer is to create sustainable tests. How can Selenium tests be made more maintainable?

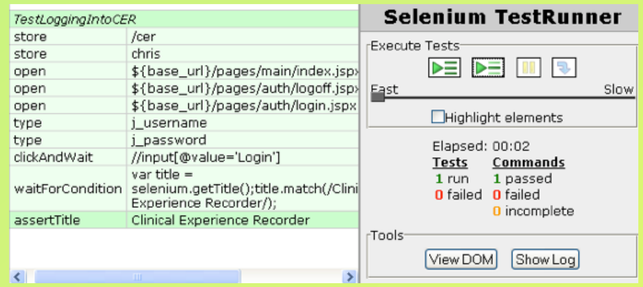
Why Automate Tests?

- Automation ensures that the software is being testing thoroughly and often.
- Manual testing is tedious and inefficient.
- When modifications are made, Automation ensures that the same checks are being made.
- Scripting conveniences such as loops and data storage come in handy.

Manual Testing:

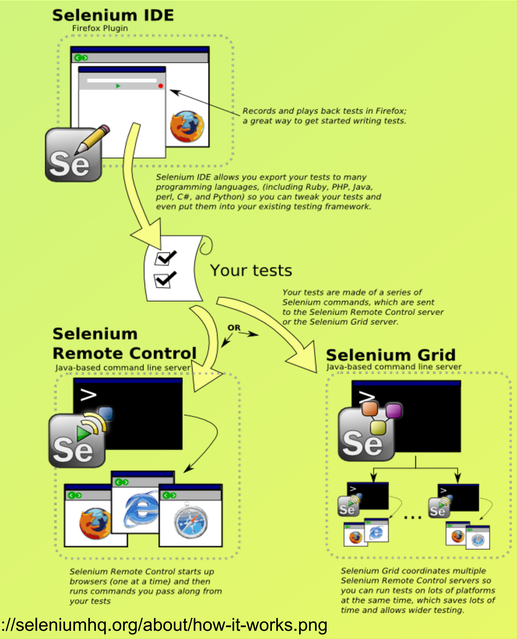
Browsing the app., over and over…

Automation:



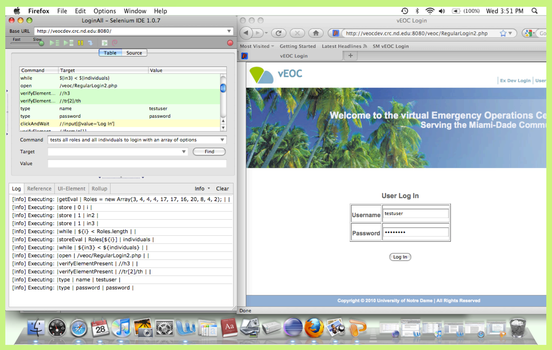
Determines functionality with the click of a button!

Cross-Platform

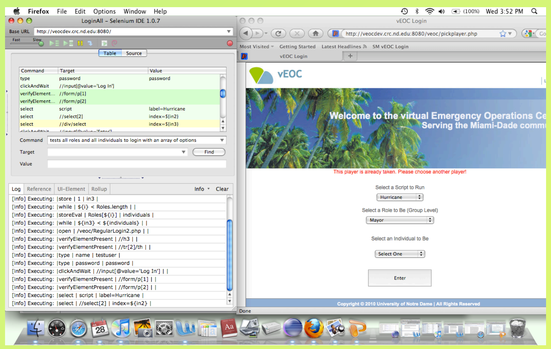


http://seleniumhq.org/about/how-it-works.png

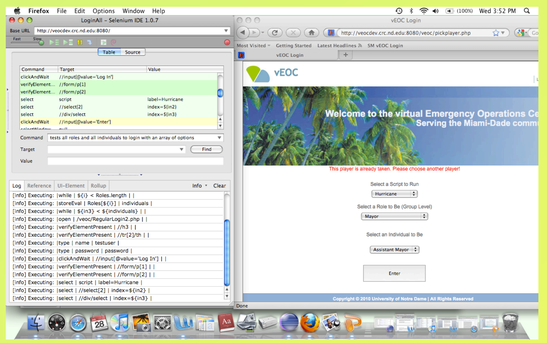
Selenium IDE



This test case is a perfect example of how to take advantage of user extensions. By implementing a JavaScript user extension, I was then able to iterate through the login options with one simple test case.



This test script is fairly maintainable. It iterates through nearly 100 options with only a few lines of commands. If the developer were to change the number of options, this test could be easily modified by changing the length of the array, instead of rewriting various test cases.



However, here, it would be great to call upon and run a different test script which could verify elements on the next page (the main panel) while the test continued. This method would make it so that the verifications could modified separately and reused elsewhere. Self-contained tests are the key to test longevity.

Locating Elements

Traditional capture/replay tools provide a basic test automation solution by recording mouse coordinates and user actions as test scripts, which are replayed to test GUI-based Applications. Since these tools use mouse coordinates, test scripts break even with the slightest changes to the GUI layout. Selenium avoids this problem by capturing values of different properties of GUI objects rather than mouse coordinates. However, an additional application must be downloaded in order to access the DOM and discover these elements. It would be best if this feature was added to Selenium.

This research was supported in part by the National Science Foundation under grants CNS-050348 and CNS-0751120